Portable Gathering System for Monitoring and Online Calibration at ATLAS.

P. Conde Muño on behalf of the ATLAS TDAQ HLT group *[1]

Abstract

During the runtime of any experiment, a central monitoring system that detects problems as soon as they appear has an essential role. In a large experiment, like ATLAS, the online data acquisition system is distributed across the nodes of large farms, each of them running several processes that analyse a fraction of the events. In this architecture, it is necessary to have a central process that collects all the monitoring data from the different nodes, produces full statistics histograms and analyses them. In this paper we present the design of such a system, called the gatherer. It allows to collect any monitoring object, such as histograms, from the farm nodes, from any process in the system, either software or hardware. The majority of users, however, change the protocol used for the communications in a system using different protocols at the same time or just publish single event quantities. The monitoring system for the ATLAS experiment has several test beams. An evaluation of this first prototype has been implemented for ATLAS and has been running during this year’s combined test beam. An evaluation of this first prototype will also be presented.

REQUIREMENTS

The monitoring system for the ATLAS experiment has very strong requirements, focusing on user friendliness and flexibility. A scheme showing the general idea of the system is depicted in figure 1. From the user’s point of view, the gathering system should be completely transparent. Their aim is to produce the histograms or monitoring quantities using their standard software environment and tools, without worrying about the data transfer or the communication protocol. From the point of view of the gatherer itself, the requirements can be summarized in the following way:

- It should be able to work in push or pull mode. This means that either the gatherer is waiting until the different processes send the monitoring data to it, or it requests the data from each one of the processes that are producing monitoring information.
- For each of the monitoring variables, it should be possible to choose if the gatherer has to sum up statistics or just publish single event quantities.
- The gatherer must be capable of using different communication protocols, being able to receive data using one protocol and send it using a different one. It may also be possible to receive data from different sub-systems using different protocols at the same time or change the protocol used for the communications in a new run.

1 Users here are any developer that wants to monitor something on the system, either software of hardware. The majority of users, however, come from the software developers community.
Figure 1: Scheme of the monitoring system for ATLAS. Any process in the Trigger DAQ chain can produce monitoring information that is distributed between the nodes of the corresponding farm. A central process, called gatherer, collects all the monitoring information, adds up statistics and sends the full statistics monitoring data to the online display and archiving. It provides also the infrastructure for the intelligent monitoring, that produces alarms and data quality assessments.

- The gatherer should react to state transitions (start/stop of run, configure, unconfigure, or any other one) since it should take certain actions at given times, like saving the histograms at the end of the run.
- It should be possible to send/receive not only histograms but also any user defined data structure.
- The gatherer should have a dynamic configuration, in such a way that new histograms or monitoring quantities are automatically taken into account, without the need of modifying the system configuration.
- The monitoring system should provide the framework for a more intelligent monitoring, executing user-defined algorithms that further analyse the monitoring data to produce alarms and data quality statements or new monitoring quantities, more meaningful and easier to understand for the shift crew.

**DESIGN**

The monitoring system can be divided in two very different parts: the first one corresponds to the infrastructure that makes the gatherer an online application to be run at the DAQ system of ATLAS. The other one consist of all the software that deals with the communications and the manipulation of the monitoring data. In the two following subsections the design of each of this parts will be explained in more detail.

**Data transfer and communications**

The design of the software to deal with the monitoring data and the communications has been done taking advantage of object oriented techniques, using the UML modeling language, since the implementation of the monitoring system had to be done in C++.

In order to fulfill all the requirements, each monitoring object should be handled in an independent way. Therefore, it is logical to define an object whose main purpose is to keep a copy of the monitoring data and deal with it. This object is called `MonObject` and it plays a central role in the monitoring system. According to the second and third requirements, it must be possible to configure each `MonObject` in a different way so each piece of monitoring data is treated according to its specifications. Thus, each `MonObject` configures itself at creation time.

To perform the data transfer two other objects were defined: a communications client object and a server object. They are tools that the `MonObject` uses to send or receive data whenever it is necessary. They have been designed as abstract interfaces, that can be implemented in many different ways according to the different protocols, but are always handled in the same way. Since the `MonObject` does not know about communication protocols, the server and client objects are created through a factory class pattern[3]. In this way, all the code dealing with communication protocols and data transfer is hidden inside the implementation of the server and client objects, and in the factory class. This allows to easily introduce new protocols without mod-
The gatherer reacts to state transitions, receiving orders from a controller. At the end of the run, for instance, the data is collected and it is sent to the archiving system.

**IMPLEMENTATION: FIRST PROTOTYPE**

The first prototype of the monitoring system has been implemented for the ATLAS combined test beam this year (summer 2004). The focus has been to provide a robust and user friendly monitoring system. The implementation was simplified with respect to the design due to the time constraints: the gatherer was configured only at start up, providing minimal interaction with the users once started (i.e., sending commands to the gatherer was not allowed), and only one protocol was implemented since there was no need to have more.

The communication protocol used is the standard one for the online applications at ATLAS, called Information System (IS)[2] and based on CORBA. In IS, all the information is stored in a central repository server. Each process sends its monitoring data to an IS server and the gatherer gets it from there. This method avoids slowing down the online processes by having too many requests coming from other processes.

To simplify the task of the software developers, an algorithm is executed at the Event Filter (the last trigger level) after all the reconstruction and monitoring packages. It gets the monitoring data, that in this case is only histograms, from the process memory and sends it to the gatherer. This procedure ensures that all the communications and data transfer is completely transparent for the users. It also ensures that new histograms are automatically received from the gatherer, without changing the databases.

Other processes from the Trigger DAQ[4] chain by default publish their information in one of the IS servers and therefore the users do not have to do anything special to get their information collected and summed up by the gatherer.

There is only one restriction concerning the name of the published in-
**Performance**

The gatherer has been running stably at the test beam since the beginning of August. Due to the fast evolution of the test beam software, new algorithms were added very frequently, so the average number of histograms and information transferred was increasing with time. The final size of data transferred was about 90 MB when the full Event Filter farm was running. In 2007, when ATLAS will start the commissioning, it is expected at least a factor 10-100 larger data size, since some of the detectors are planning to fill occupancy plots per detector channel\(^3\) and the size of the Event Filter farm will also be larger.

The CPU consumption during the test beam was about 30\%, in a Xeon machine with a 3.2 GHz processor and 1 GB of memory, when the data is being transferred. The gatherer is idle about 30\% of the time, waiting some time before collecting and sending the data. The memory consumption was about 10\%, also during data transfer.

Since the IS server is not a real time system, when there are many processes sending large amounts of data at the same time, the data transfer may become slow. The flexible configuration of the gatherer, however, allows to improve the performance of the system by simply distributing the data within more servers or introducing more gatherers running in parallel. A tree of gatherer processes could be introduced if necessary.

**SUMMARY AND CONCLUSIONS**

The first prototype of a gathering system for the ATLAS monitoring has been designed and implemented for the ATLAS 2004 combined test beam, and it has been running stable for more than one month. The system has been designed to have maximum flexibility and user friendliness. It abstracts the communication layer, allowing transparency for the user and the possibility to use different protocols for the data transfer. It also makes the gatherer easily adaptable to any new experiment.

Having dynamic configuration and the possibility to use different communication protocols allows to tune the system in order to optimize the performance during the data taken.

**REFERENCES**


\(^3\)There will be of the order of few hundreds million channels.